IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 13, 2020, accepted December 28, 2020, date of publication January 11, 2021,
date of current version January 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050670

Efficiency Versus Accuracy: A Review of Design Techniques for DNN Hardware Accelerators

CECILIA LATOTZKE AND TOBIAS GEMMEKE, (Senior Member, IEEE)
Chair of Integrated Digital Systems and Circuit Design (IDS), RWTH Aachen University, 52074 Aachen, Germany
Corresponding author: Cecilia Latotzke (latotzke@ids.rwth-aachen.de)

ABSTRACT Deep neural networks (DNNs) have surpassed other algorithms in analyzing today's abundant data. Due to the security and delay requirements of the given applications, analytical data extraction should happen on edge devices. Edge devices, however, struggle to support the increasingly complex DNNs because of the models' high computational load and number of employed parameters. Thus, edge devices need support by efficient accelerators to process DNNs. However, the design of DNN accelerators remains challenging, as there is a lack of established design techniques directed towards a specific design point in terms of energy budget, area, time-to-solution, and classification accuracy. This article fills this gap by providing a quantitative, large-scale, state-of-the-art comparison of DNN accelerators building on published data subjected to technology scaling and benchmark normalization. The leveled comparison stimulates learning from previous designs by considering the impact of each technique on energy, area, and time-to-solution. Furthermore, the key design techniques used in the DNN accelerators are classified according to their influence on the classification accuracy. Finally, we provide a discussion of hardware accelerators to support future designers in considering the trade-off between efficiency and accuracy in order to identify the most suitable techniques for certain benchmarks.

INDEX TERMS Neural network hardware, artificial neural networks, multi-layer neural network, image classification, integrated circuit technology, solid state circuits.

I. INTRODUCTION

The available amount of data that has to be analyzed is steadily increasing. In many fields, the analytical information extraction requires high security and low latency [1], [2]. Hence, the tasks should ideally be solved where the data is generated, i. e. on edge devices. In many domains, deep neural networks (DNNs) are a preferred choice because of their excellent accuracy; however, they come at a high price in terms of multiply-and-accumulate operations (MAC) and memory footprint. With increasing capabilities of such DNNs, edge devices lack resources to meet throughput requirements at their available energy budget.

This shortcoming drives the quest for hardware solutions which support the execution of advanced machine learning algorithms on the edge. FPGAs are well-suited for related design space exploration with their beneficial combination of short design cycles and full flexibility in terms of the emulated micro-architecture. Various surveys have addressed this domain including the manifold software and hardware techniques focusing on performance [3]–[7]. Considering dedicated ASIC implementations, one has to keep in mind their significant differences compared to FPGAs [8], such that key findings in one domain might not be applicable to the other. However, studies that review ASICs are – to our knowledge – scarce and limited in scope.

General algorithmic optimization as well as hardware techniques for DNNs on ASICs were investigated in [9] and [10]. The sketched optimization methodology in [9] includes software, hardware, and run-time optimization. Emphasis is put on the understanding of the data flow that includes both on-chip and off-chip memory access. Proper management of the data movement is fundamental to designing an efficient hardware architecture from a system perspective. This is stressed by the energy for off-chip accesses, e. g. one DRAM access is $\sim$200$\times$ higher than the computational energy for a standard MAC [11]. A classification of the data flow and its optimization are presented in [10], referring to systolic principles [12], and further in-depth analyses of the cost of a given hardware accelerator, i. e. energy and time-to-solution, were performed using an automation tool [13]. This tool aims at providing a fine-granular breakdown of the efficiency

The associate editor coordinating the review of this manuscript and approving it for publication was Yasar Amin.

and performance of a given hardware architecture for a specific workload. These works either grouped and summarized design techniques for hardware accelerators or provided detailed analysis of selected references using complex tools.

In contrast, this article focuses on the question of how much an individual design technique contributes to the final energy efficiency and performance of hardware accelerators for DNNs. Our work specifically assesses hardware architectures and their key techniques on the basis of scaled figures of the published raw data. Our main contributions include:

- specific cost functions to evaluate hardware accelerators for DNNs (Sec. II-C),
- a methodology for quantitative comparison applying scaling to a reference technology (Sec. III-A) as well as normalizing the data points provided in the analyzed publications w. r. t. the complexity of used dataset, word length, and DNN model (Sec. III-B), a quantitative assessment and common trend of state-of-the-art hardware solutions according to the cost functions for hardware accelerators (Sec. III-C),
- a dissection of the convoluted use of design techniques, analyzing their individual cost and benefits (Sec. IV-A and (Sec. IV-B),
- a summary of applied techniques regarding the achieved efficiency vs. accuracy trade-off (Sec. IV-C),
- and a design space exploration for the application of incremental optimization steps of orthogonal techniques to an already remarkable design (Sec. V).

II. BACKGROUND

The following section introduces the terminology used in the context of DNNs and their hardware acceleration. Starting with commonly addressed classification tasks and benchmarks, a general introduction to DNNs is given, including quantitative metrics for the assessment of neural network models, hardware architectures, and DNN accelerators.

A. DATASETS

DNNs are currently applied to a multitude of classification tasks in handling images and audio as well as other time series as common in the medical domain. In order to base our work on a larger pool of proposed solutions, we focus on the most commonly used benchmarks to extract the trade-offs between efficiency and accuracy of different design techniques. The most commonly used benchmarks for hardware accelerators are the following: the Modified National Institute of Standards and Technology database (MNIST) [14], the Canadian Institute For Advanced Research (CIFAR 10) dataset [15], and the ImageNet Challenge [16].

MNIST is a low-complexity example, which contains 60 thousand black and white images with a size of 28×28 pixels. The images contain handwritten digits from 0 to 9. The CIFAR 10 comprises 60 thousand RGB images with 32×32 pixel. The '10' in CIFAR 10 signifies the 10 used classes. The ImageNet Challenge includes 1.2 M images, which vary in size and are grouped into 1000 classes. The error rate is classified either as a top-1 score when only considering the dominant classification result, or as a top-5 score, which checks the 5 labels with highest probability against the expected label.

B. NEURAL NETWORK MODELS

In the literature, both, neural network model and neural network architecture, are used to describe the structure and applied operations. Here, we refer to (neural network) models and (hardware) architectures. A model consists of several layers. The input size of the first layer is identical to the dimensions of the input data. Inputs to a layer l are called activations. The classification layer is the last layer in a neural network. Such neural network models can be differentiated in terms of number, type, and connectivity of its layers. Preceding the DNNs, a multilayer perceptron (MLP) consists of only a few fully connected (FC) layers. All d^l input activations of an FC layer are multiplied with d^{l+1} weight vectors ${w_i}^l$ (dot product) resulting in a vector of output activations with d^{l+1} elements. The costs for a FC layer are summarized in Table 1.

TABLE 1. Cost for CONV and FC layers.

Cost	FC layer	CONV layer
weights	$d^l \cdot d^{l+1}$	$k_\mathrm{H} \cdot k_\mathrm{W} \cdot d^l \cdot d^{l+1}$
output activations	d^{l+1}	$m_\mathrm{H}^{l+1} \cdot m_\mathrm{W}^{l+1} \cdot d^{l+1}$
MAC	$d^l \cdot d^{l+1}$	$k_\mathrm{H} \cdot k_\mathrm{W} \cdot m_\mathrm{H}^{l+1} \cdot m_\mathrm{W}^{l+1} \cdot d^l \cdot d^{l+1}$

Whereas FC layers utilize one dimensional vectors, convolutional (CONV) layers follow the introduced principle, but for three dimensional matrices. CONV layers form the basis of so called convolutional neural networks (CNN), the most commonly used DNNs for image classification. To extract features anywhere in an image, d^{l+1} convolutional kernels of width k_H^l, height k_W^l and depth d^l are applied in the CONV layers. Thereby, d^l input channels are mapped to d^{l+1} output feature maps of height m_H^{l+1} and of width m_W^{l+1}. The corresponding computational cost is detailed in Table 1.

The research in CNNs started with [17] but it took until 2012 for DNN to take off with the success of AlexNet in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012 [18]. It is the most commonly referred model mapped to hardware accelerators to classify the ImageNet dataset.

C. COST FUNCTIONS

DNNs are usually compared on the basis of their classification performance, i. e. the achieved accuracy or error e, which can be used interchangeably as in equation (1).

$$e = 100\,\% - \text{accuracy} \tag{1}$$

Models of comparable accuracy are further assessed w. r. t. memory requirements and arithmetic complexity in terms of MAC. The memory footprint is driven by the number and word length of stored values such as weights and activations.

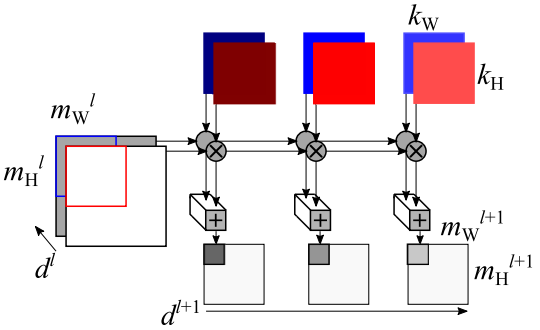


FIGURE 1. Schematic of operations in a convolutional layer.

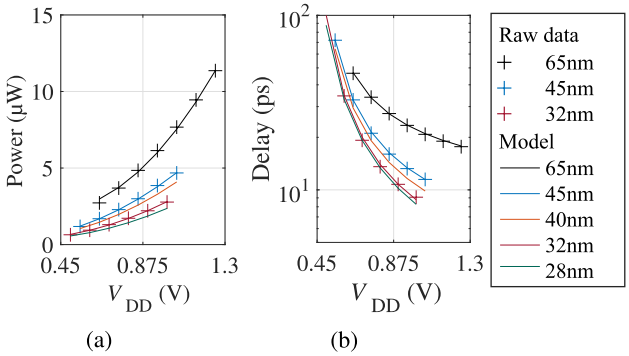


FIGURE 2. Scaling of (a) power and (b) delay with supply voltage V_{DD} using model equation (2) and respectively equation (3) (both are fitted to raw data of [19]).

Typically, benchmarking of ASICs of similar functionality lists throughput in terms of GOps/s, energy efficiency in TOps/s/W and area efficiency in GOps/s/mm^2. For an apples-to-apples comparison, any such data should be normalized to the same technology and supply voltage.

Considering a specific application, these metrics provide unsatisfactory results as there can exist a significant gap between actual throughput and theoretical peak performance. Hence, for the assessed references, we compared the metrics energy E in µJ/image, area A in mm^2 and time-to-solution T in s/image. The latter indicates the time from applying a pattern to the hardware accelerator to the moment the classification result is computed. The product of area A and time-to-solution T is called AT complexity.

III. PRESENTATION OF REFERENCE DATA

As stated above, the following benchmarking will rely on normalized data points. For this purpose, we scale reference data to the same technology and normalize to the anticipated baseline complexity.

A. TECHNOLOGY SCALING

The manufacturing technology has a decisive impact on key performance metrics of an ASIC design. To evaluate the benefit of the various design techniques in a manufacturing technology agnostic fashion, the reported area, power and speed has to be normalized accordingly. To bound the introduced inaccuracy in such a step, we used the extracted raw data of [19] to fit to each technology node a second order model to estimate energy per operation (cf. equation (2)) and an alpha-power-law based model of propagation delay (cf. equation (3)) as function of supply voltage.

$$S_{\text{power}} = a_{\text{p2}} \cdot V_{\text{DD}}^2 + a_{\text{p1}} \cdot V_{\text{DD}} + a_{\text{p0}} \tag{2}$$

$$S_{\text{delay}} = \frac{C_{\text{eff}} \cdot V_{\text{DD}}}{I \cdot (V_{\text{DD}} - V_{\text{th}})^{\alpha}} \tag{3}$$

$$S_{\text{area}} = \frac{\lambda_{65\,\text{nm}}^2}{\lambda_{\text{ref}}^2} \tag{4}$$

$$x_{\text{scaled to 65 nm}} = x_{\text{ref}} \cdot \frac{S_{65\,\text{nm}}}{S_{\text{ref}}} \tag{5}$$

Both models fit the raw data well as highlighted in Fig. 2. Because there is no raw data given in the reference for 40 nm and 28 nm, the corresponding fitting factors were derived by interpolation from the raw data of the high-k technology nodes as used in [19]. The area scaling factor S_{area} simply follows the square of the technology node. The scaling of power, delay, and area is done according to equation (5), where the baseline is a 65 nm technology operating at 1.1 V nominal voltage (cf. [19]).

B. BASELINE COMPLEXITY

The number of MAC per classification is a function of DNN model with a higher MAC number loosely correlating with better accuracy. Despite the dominant use of MAC as quantitative metric in comparisons, the actual movement of a single word can feature a 200× higher energy usage [11]. Therefore, our benchmarking accounts for the number of off-chip memory accesses required by the hardware architecture. If the total number of off-chip accesses are not specified in the publication, we assume a minimum number of off-chip accesses, equal to the count of weights and elements of the input image.

At the end, the quantitative comparison should be agnostic concerning the complexity of the neural network model as we intend to assess the efficiency of the hardware realization. Hence, we normalized AT complexity and energy E with respect to the baseline complexity of the underlying model to a single MAC. This requires an estimate of the energy and AT complexity as functions of the adopted word length for weights and activations. For this purpose, we refer to a vector multiplication described in HDL and synthesized in a commercial 22 nm FDSOI technology at 0.8 V with word length varying from 1 to 32 bit [20]. Related technology scaling is applied as described above. In summary, we calculated normalization factors specific to each design point as a function of applied DNN model and word length.

C. PARETO PLOT

The Pareto plot in Fig. 3 shows the correspondingly normalized metrics of all considered references. The normalized AT complexity on the x-axis is shown vs. normalized E on the y-axis. Hence, the location in the 2D plot reflects the ATE

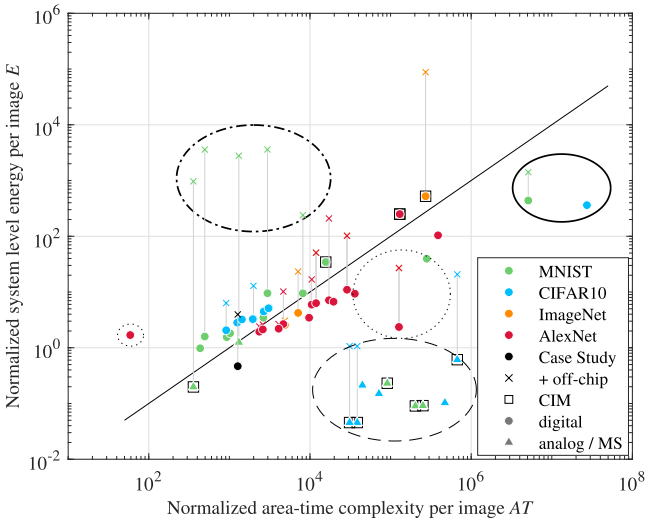


FIGURE 3. Energy and *AT* complexity per MAC for all benchmarks. The black circle indicates the spiking hardware with huge DNN, the dotted circle indicates the bit-serial processing designs, the dashed circle indicates the analog and CIM designs, and the dash-dotted circle indicates the MNIST MLP designs with added off-chip access.

complexity with efficiency increasing towards the origin. For better comparison, colors differentiate between the datasets MNIST, CIFAR 10, and ImageNet.

The symbols highlight the applied design style differentiating between digital and analog and mixed-signal (MS). The analog (and MS designs) are not scaled [21]. Furthermore, a box indicates that the implementation adopts a computing-in-memory (CIM) architecture. Lastly, the designs are classified by whether they require off-chip access, as the set of weights do not fit into on-chip storage. If in such cases the energy of reloading is not accounted for, we indicate its estimate with an '×', whereas we use for all designs DDR3 as off-chip memory technology. Thereby *AT* complexity stays constant.

If the weights for the biggest CONV layer cannot be kept on-chip, tiling would be necessary. Tiling means that only a part of the input feature map and / or of the filter can be kept on chip to process a partial sum of the output pixel. Thus, it is necessary to reload elements, which leads to an increased number of off-chip accesses. Because the tiling concept is often not explicitly stated, only data points are considered which either can store all weights of the biggest CONV layer simultaneously or state their total number of off-chip accesses. Consequently no tiling is applicable.

Guided by the location in this design space, the designs are analyzed in the following section with the objective of extracting essential design techniques, pointing towards highly efficient design techniques.

IV. ANALYSIS OF DESIGN TECHNIQUES

The following sub-sections describe the various design techniques found in literature. The presentation includes a discussion of their impact on energy E, area A, and time-to-solution T. The discussion is split into techniques with impact on accuracy (Sec. IV-A) and those without impact on accuracy (Sec. IV-B). As reference, all techniques are summarized in Table 3.

A. TECHNIQUES WITH IMPACT ON CLASSIFICATION ACCURACY

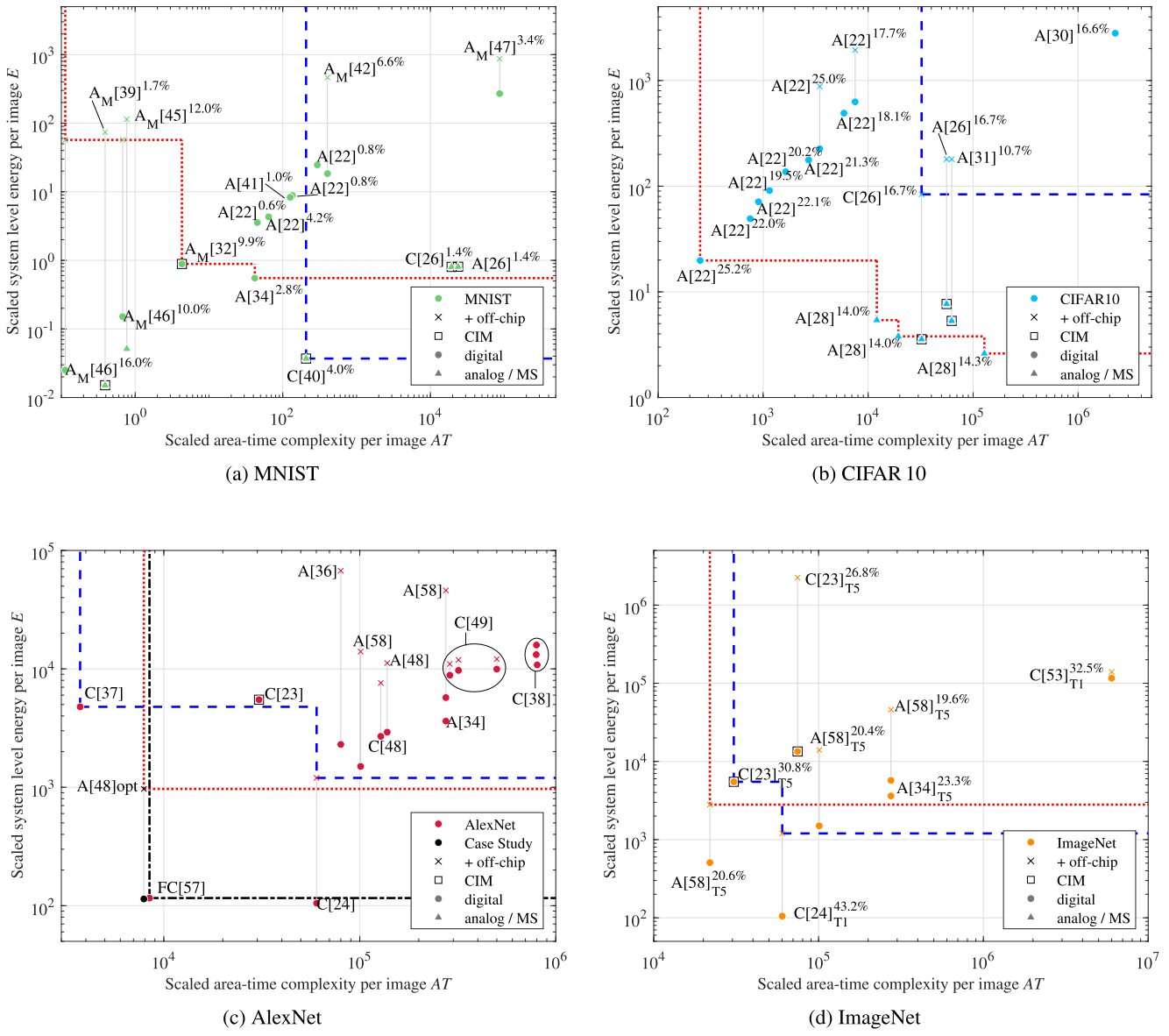
1) CHOICE OF DNN MODEL

In a top-down fashion, the choice of the neural network model is a fundamental design parameter to trade accuracy for efficiency. In a first rush, network complexity followed the trend of hardware providing ever increasing performance. Today, research derives more efficient models that achieve high accuracy, while requiring less operations.

Fig. 4a, Fig. 4b, and Fig. 4d depict accelerator designs for the most commonly used benchmarks MNIST, CIFAR 10, and ImageNet. These figures contain a diverse portfolio of DNNs. Unfortunately, the number of publications on lightweight DNNs, like MobileNet or EfficientNet, is too limited for a quantitative comparison. Therefore, Fig. 4c consolidates accelerator designs for AlexNet. The individual data points are labeled according to their corresponding reference. Each data point is appended with additional information. Starting from left to right, firstly, the plots highlight whether a datapoint includes all necessary layers 'A', or is limited to either the fully-connected 'FC' or convolutional 'C' layers, only. This differentiation is necessary, because an accelerator design which is capable of processing efficiently all layers in a DNN is more flexible than a highly specialized accelerator for only a few DNN layers. The index 'M' is relevant for MLPs, because here all layers is equivalent to all fully connected layers, due to the structure of MLPs. Secondly, the label includes the resulting error in percent. The subgraph (c) lacks this metric as the references did not refine to the individual classification accuracies. Thirdly, the symbols highlight the applied design style and benchmark as in Fig. 3. For each group, the Pareto optimal front, with regards to the *ATE* complexity, is limited to data points including the total system energy.

CONV layer have the potential of much higher computational intensity than FC layer, as the former can reuse the filter kernels many times. An example for this effect is the MNIST benchmark: here especially MLPs suffer extensively from off-chip access, because they use each weight once and have a reduced number of MAC. Normalized according to the computational complexity, Fig. 3, this effect is even more prominent. Because of these limitations, MLPs are in practice not applied for more complex benchmarks.

Fig. 4a-d shows an increase for *AT* complexity and energy E for increasingly complex benchmarks. This is correlated by the increase in DNN size, which causes an increase in memory footprint and especially for CNNs, an increase in MAC, cf. Table 1. Reduced number of operations should proportionally decrease processing time and computational energy to the first-order, while classification error is increased.



(a) MNIST (b) CIFAR 10

(c) AlexNet (d) ImageNet

FIGURE 4. Pareto plot for hardware accelerators for four different benchmarks, (a) MNIST, (b) CIFAR 10, (c) AlexNet, and (d) ImageNet.

However, this hypothesis is contradicted by [58], with 11× less MAC than [53]. The former design uses MobileNet, whereas the latter uses VGG 16. Hence, the efficiency and accuracy benefit of the use of more recent DNN models.

2) QUANTIZATION

The development of models is typically done by using high precision numerics to achieve best accuracy, excluding side effects related to limited precision. A variety of half precision floating point formats has been introduced to optimize training and improve efficiency of software oriented implementations. In the case of efficient ASIC implementations, the floating point numbers, as used during training, are typically converted in a first step to a fixed point representation.

Many studies have shown that *reduction of word length* can be done to a certain degree without impacting accuracy. The proportional savings in required memory bandwidth and memory footprint, as well as the significant savings in the computation of the MAC, have led to many works pushing word length reductions even further. In the end, the combination of reduced word length and fixed point representation induces a major reduction in dynamic range and precision as compared to the floating point values. Furthermore, the choice of a word length and selection of radix point fixes the trade-off between precision and range.

In the extreme case, quantization results in binary representation that uses a single bit to differentiate between $+1$ and -1. Two major benefits of this technique are the minimal memory requirement per weight and activation, and the

mapping of the multiplication to a binary XNOR operation. The latter resulted in highly optimized datapaths dedicated to binary DNNs [23]. The former benefit is proportional to the word length reduction. It is exploited for ImageNet and AlexNet in [23]–[25], for CIFAR 10 in [26]–[31] and for MNIST in [26], [27], [31]–[35].

In general, uniform word length reduction leads to reduced classification accuracy. Better results are achieved by *independently tuned* representation of weights and activations. For the case of AlexNet, there are approx. 65× more weights than activations. Thus, word length reduction for weights has a higher impact on the memory footprint. At the same time, a loss in accuracy can be mitigated by preserving higher word lengths for activations as they convey the contained information between layers. Using a relatively larger word length for activations than for weights has the potential to provide either lower cost at same accuracy or better accuracy at same cost, when compared to a design with uniform word length. This technique was applied for AlexNet in [36]–[38], for CIFAR 10 in [22], [31] and for MNIST in [22], [39], [40]. It is worth noting that for MNIST this technique improves classification accuracy compared to equal word length for weights and activations, cf. [22].

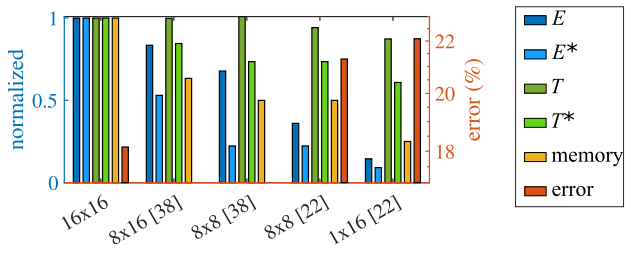


FIGURE 5. Reduction of word length on energy E, time-to-solution T and memory footprint is shown on the primary y-axis. Ideal scaling assumption is indicated with '$*$'. Normalization refers here to the corresponding 16 bit × 16 bit design point for each reference. Word lengths of weights w, in bit, and activations a, in bit, of the references are indicated as $w \times a$. The impact on error is shown for CIFAR 10 [22] (no data for the case of AlexNet in [38]).

The correlation between energy, time-to-solution, and memory footprint depends also on the application and underlying hardware. Fig. 5 illustrates the trade-off for energy and time-to-solution for both quantization methods: identical and different word lengths for activations and weights. The work of [22] indicates that reductions in word lengths causes an increase in classification error, as 16 bit × 16 bit achieves an error of 18.14 %, 8 bit × 8 bit achieves an error of 21.3 %, and 1 bit × 16 bit achieves an error of 22.1 % for CIFAR 10. The examples in Fig. 5 were chosen because they follow the 'ceteris paribus' principle of an ablation study varying only one parameter at a time. Thus, any efficiency gain can directly by related to the chosen quantization. If the word length is fixed prior to designing the chip, the memory footprint can be reduced significantly. Hence, the necessary total area of the chip can be reduced.

A classic solution with moderate effort and efficiency is adapting a fixed precision multiplier to support multiple parallel multiplications of sub-divided word lengths. For example a 16 bit × 16 bit multiplier could be used for two 8 bit × 8 bit or four 4 bit × 4 bit multiplications. Ideally, this processing mode could reduce energy and time-to-solution by 4× and 16×, respectively. At the same time, the overhead of this processing mode has to be carefully weighed as it requires additional registers to capture intermediate results and control of iteration count across the word length.

Assuming a dedicated implementation, the energy reduction for 8 bit × 16 bit and 8 bit × 8 bit compared to the baseline of 16 bit × 16 bit should be 53 % and 22 %, respectively [20]. However, [38] achieves a respective energy reduction of only 84 % and 68 %. A striking contrast to this design is [22], which achieves for 8 bit × 8 bit 36 % of its baseline energy. It is worth mentioning that [38] was taped out but [22] was not. The inefficiency of [38] for lower word lengths is underlined by the effect on time-to-solution, which is expected to reduce by 85 % and 75 %, respectively, but does not decrease at all. Thus, it seems as [38] is optimized for 16 bit × 16 bit. In general, higher flexibility comes with the price of reduced efficiency [38], [41], [42]. For this reason, most hardware accelerators for DNNs support only a limited number of DNNs very efficiently.

Going one step further, DNNs feature layers of varying complexity in terms of number of feature maps and weights. Also, the sensitivity towards quantization noise is a function of the analyzed layer. It follows, that for a certain accuracy goal the word length requirement varies throughout the model. This fact can be exploited by *layer-wise quantization* while maintaining classification accuracy as was done in [24], [25], [36], [37]. Besides appropriate data formats, the functional units have to be designed to support varying word lengths to maximize the benefit.

A benefit of a linear *quantization pattern* is its relative ease of use. Other quantization techniques have been proposed to improve the represented dynamic range as well as to provide higher precision for values close to zero. One option is logarithmic quantization as in [34], [35]. Here, linear quantization, Q(), is applied to the binary logarithm of the weights w, and activations a:

$$w_{\mathrm{Q}} = \mathrm{Q}(\log_2(w)) \tag{6}$$

$$a_{\mathrm{Q}} = \mathrm{Q}(\log_2(a)). \tag{7}$$

In the extreme case, the number is quantized to the first non-zero digit reducing multiplications to binary shift operations.

Finally, another advantage of strong quantization is the increase in sparsity, if small values are set to zero. This increased sparsity in turn is exploited by other techniques such as compression or zero-skipping. Furthermore, quantization reduces the amount of representable values, which is beneficial for techniques like weight sharing.

3) ANALOG AND MS DESIGN STYLE

ASICs can be designed in digital, analog, or MS design style. Key motivation to scale according to Moore's law [43] is cost cutting by reducing the size of SRAM bit-cells and increasing the gate density. At the same time, scaling improves performance and energy efficiency. In the case of analog design, size reduction has come to a halt [21] as uncertainties increase inversely with the feature size. Even in older technologies, 8 bit resolution has been considered the break-even between analog and digital signal processing [44]. This implies, that low precision operations can be realized more efficiently adopting an analog design style.

Yet, all analog and MS design suffer from accuracy reduction compared to their reference classification accuracy, e.g. [26], [27] suffer from 0.2 % (CIFAR 10) - 0.3 % (MNIST) and [39] suffers from 0.5 % (MNIST) - 5 % (CIFAR 10), due to inherit reduction of the SNR for these designs. This SNR reduction can have different additive causes, like quantization, thermal effects, or voltage scaling cf. Sec. IV-B7. For [28], [29] SNR reduction is limited to voltage scaling, due to the 'ceteris paribus' principle. Hence, lower energy and faster time-to-solution are possible at the cost of increased classification errors.

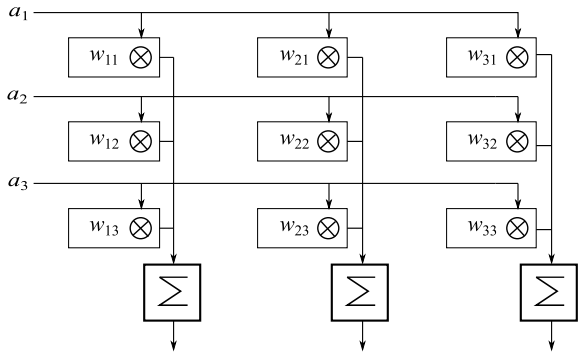


FIGURE 6. Fan-in based data flow: stationary weights *w* in bit-cells, broadcast of activations *a* and fan-in of partial sums.

A comparison between both design styles shows that MS designs are more common [26]–[29], [39], [45] than purely analog ones [31], [40]. Both target mostly small benchmarks, like MNIST and CIFAR 10 (cf. Fig. 3). The designs with a significant lower energy than the trend line, in Fig. 3 encircled with a dashed line, share the same data flow, cf. Fig. 6. Hence, this data flow assures low energy per operation while requiring significantly higher *AT* complexity. The data flow is mostly used by CIM implementations, but not limited to these. The weights are stationary inside the SRAM and the input activations are broadcasted horizontally to the bit-cells. Inside the bit-cell, the partial sum is computed by multiplying the weight with the input activation. These partial sums are accumulated vertically via fan-in to feed the output activations. Most benefit is realized by mapping the accumulation to an integration of charge on the bit-line of a memory, giving rise to the many analog or MS designs incorporating CIM.

4) ANALOG COMPUTING-IN-MEMORY

CIM generally refers to the concept of overcoming the memory wall by executing the computations directly at the storage locations. In a broader sense, the memory periphery or circuits directly attached to the memory compute the results. The narrower interpretation places operations within the bit-cell array. Extreme cases are cross-bar arrays, which utilize memristive switching devices to store a weight value persistently. Due to a lack of details, such as specifying classification accuracy, no memristive design are included. Therefore, cross-bar arrays are not further considered in the following.

Most analog CIM designs map storage of weights and computation of partial sums to the same memory [26], [27], [31], [39], [40]. This occurs usually in combination with low bit-width from 8 bit down to binary. The partial sum is accumulated as charge on the bit-line. The bit-line is directly charged via either a current for a certain duration [39], [40], or a bit-cell capacitor [26], [27], [31]. The total system energy E per analog CIM operation varies widely from 14.3 fJ / bit [39] to 2.5 pJ / bit [26], [27]. A comparable digital design [34] achieves for the same benchmark, MNIST, 12.6 pJ / bit. Hence, analog CIM improves energy efficiency.

5) SPIKING NEURAL NETWORKS

Following more closely the operational principle of the human brain, spiking neural networks (SNN) have the promise to achieve higher energy efficiency than conventional computing. In this work, spiking hardware architectures are only analyzed in their ability to support DNNs. Dedicated spiking hardware architectures designed for specific DNNs [45], [46] achieve comparable *ATE* efficiencies as their non-spiking counterparts, but their classification accuracy is 5 % or more below expectations for the given benchmark.

However, it is also possible for spiking hardware accelerators to achieve competitive accuracies, e.g. [30], even though this capability comes at the price of 76× higher normalized energy E and 277× higher normalized *AT* complexity than average. The *ATE* trade-off is below the trend line in Fig. 3. This fact indicates that the design has selected an atypical trade-off between energy and *AT* complexity. Furthermore, these architectures are conceived for a different use case - the simulation of large scale biological neural networks [30], [47]. So, a direct mapping of DNN architectures to SNNs appears not to be advantageous. To fully exploit the benefits observed in nature, more capable SNN models first have to be conceived that closely adhere to the neuromorphic principles.

B. TECHNIQUES WITHOUT IMPACT ON CLASSIFICATION ACCURACY

1) MEMORY HIERARCHY

Designers differentiate between *on-chip memory* and off-chip memory, as depicted in Fig. 7. The on-chip memory is either located inside the logic (as memory level 1) or outside of the

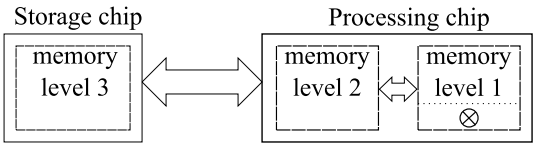


FIGURE 7. System level view for general ASIC. Level 1 and level 2 are on-chip memory and level 3 is off-chip memory.

logic (as memory level 2). Usually memory outside the logic is used to store all types of data. Hence, it is referred to as the global buffer. Memory inside the logic stores data directly required for the computations, and the granularity of memory level 1 comes in many flavors. It can be split into dedicated buffers, register-files, or even down to individual registers integrated into the PE. The off-chip memory (memory level 3) is, as the name suggests, located outside of the chip and as such not considered in the quoted area.

The energy for an off-chip memory access is approx. 130× higher [11] than the energy for an on-chip memory access. Furthermore, time-to-solution T is affected by the memory throughput and latency. Off-chip memory, e. g. DRAM, needs typically multiple cycles to provide the first word to the PE, whereas on-chip memory, e. g. SRAM, can deliver it immediately. So, a choice between on-chip memory and off-chip memory always includes a trade-off between area A, time-to-solution T, and energy E.

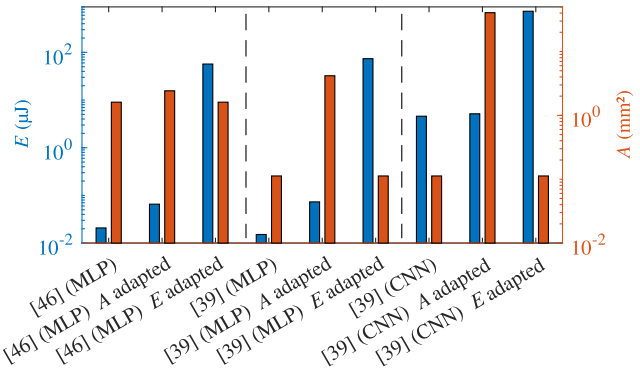


FIGURE 8. Impact of memory choice on energy E and area A, with either energy E adapted to reflect required off-chip access, or area A increased to meet the data footprint of the DNN.

To properly compare different designs, we ran a sanity check to see whether the required data fits into the on-chip memory. If the on-chip storage is insufficient, energy is added to account for the assumed minimal number of off-chip accesses to process a single image. In Fig. 8 these adjusted design points can be compared to their original counterparts. The most striking effect of this technique can be seen for [39], which uses a MLP for MNIST. Here, either energy E increases by $10^4\times$ or area A increases by $100\times$. Because the original data plus additional area would lead to an increased divergence of the original design, only the original data and the original data plus off-chip access energy are shown in Fig. 3.

The use of very small on-chip memories in combination with disregarding off-chip access is mostly seen in the MNIST benchmark for [39], [46] and in the CIFAR 10 benchmark for [39]. These architectures are not scalable to support larger networks despite promising metrics. With respect to time-to-solution and energy efficiency, an optimal design would 'simply' provide sufficient on-chip memory to fit the used DNN. Taking area into account as well, optimization has to include on-chip memory hierarchy and off-chip access as in [48]–[51] to fully exploit data reuse techniques.

The energy consumption for off-chip memory access is a product of transferred bits and energy per transfer. The latter is driven by the *off-chip memory standard*. In the following, four different off-chip memory technology options are compared: DDR3, DDR4, LPDDR3 and a 3D SRAM stack, [35].

The performance increases for each DDRx generation. Overall, most publications consider a DDR3 off-chip memory. Thus, our minimal 'off-chip data access energy' assumes DDR3 with 70 pJ / bit according to [52] as baseline. LPDDR3 was used in [53], which references 21 pJ / bit according to the LPDDR3 memory model from [54]. DDR4 was mentioned in [55] but not directly included in the energy budget. The design is customized for the read and write bandwidth of 25.6 GB/s of the DDR4-3200. According to [56] DDR4 needs 15 pJ/bit. An example for the case of 3D SRAM in [34], [35] shows that the total chip area was significantly influenced by the off-chip memory area, because the computing engine and memory are stacked. But in general, the off-chip area itself is not taken into account for the total area of a design. To write one bit via 3D SRAM TCI coil is reported at 1.6 pJ, whereas the read per bit costs 0.4 pJ.

To highlight the impact of off-chip data storage in Fig. 3 and Fig. 4, the reported computing and on-chip memory energies are supplemented with the minimal energy for necessary off-chip access. The gap between these points underlines the necessity of considering off-chip access to get the full picture. For a more advanced DDRx standard, energy efficiency and bandwidth would be significantly improved. Hence, the choice of DDRx standard influences the total *ATE* efficiency.

2) DATA REUSE

Let us consider an idealistic hardware architecture, where movement of operands and results contributes an insignificant overhead, i. e. only the energy consumption of computations counts. This perspective is highlighted as lower bound (black line) in Fig. 9. Quite apparently, all references lay above this limit, as energy used by data-movement adds to any numeric computation. Hence, increasing the computational intensity by data reuse has significant potential to increase efficiency.

Looking at the distribution, there are 65× more weights than activations in AlexNet. For CONV layers, weight reuse is obvious. Furthermore, images can be loaded in batches, whereas the number of images determines the batch size B. A batch can be processed with the same set of weights,

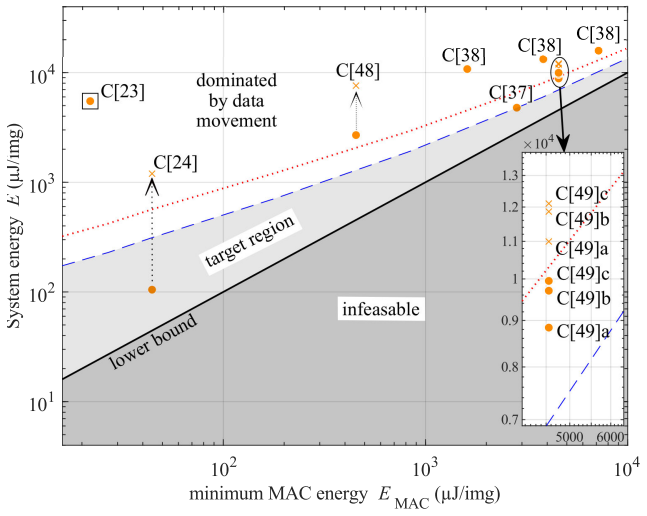


FIGURE 9. System energy vs. MAC energy AlexNet for CONV layers only (for symbols cf. Fig. 3).

called *batch processing*, and the reuse factor of each weight increases by B. For the energy efficiency focused case, i. e. without off-chip access, the energy reduction comes at a price of area increase as the on-chip memory needs to store all partial sums of the different images. As a matter of fact, the weights are stored off-chip for large networks. Thus, even an ideal architecture would have to load each weight at least once. This case is indicated as the red dotted line in Fig. 9, accounting for off-chip access as well as energy consumption for computation.

The CONV layer and, to a certain extent, also the FC layer, offer even for a single image reuse possibilities considering the computations of the output activations. Basically, the CONV layer operates over four loops of flexible execution sequence (cf. Fig. 1). The first loop iterates over the pixels in one kernel and the equivalent pixel in the feature map to be convoluted. This is most commonly known as element-wise multiplications. The second loop iterates over the input feature maps to be convoluted with their equivalent filter, e. g. an edge detection filter for a red image channel that differs from an edge detection filter for a green channel. The third loop iterates over the activations of one output feature map, thus, the filter kernel is convoluted with the complete feature map. Finally, the fourth loop iterates over the different kernel windows to step over the same input feature maps as in Fig. 1, e. g. in addition to an edge detection filter there is also a Gaussian filter. Only the second and fourth loop are used as well in FC layers. Here each activation represents an individual feature map and each kernel pixel contains a single, individual weight, which is only used once per classification. Efficient unrolling of these loops, alias *Loop unrolling*, can result in efficient data reuse schemes as shown in [23], [34], [35]. The optimal unrolling depends on the underlying hardware as well on the neural network.

Dedicated hardware accelerators, however, are not limited to loop unrolling. Because of their opportunity of an individual and fine granular memory hierarchy, they can use *data flow optimization* for further exploitation of key concepts of systolic data movement. Pure systolic arrays are basically fully pipelined networks of processing elements (PE) [12]. In this case, each PE communicates only with its direct neighboring PEs and receives and distributes data from and to them. Weights, input activations, partial products, or a combination of these can be stored within or in close proximity to the PE. The resulting data flow for these architectures can be classified according to the stored input parameter in the PE, into weight stationary (cf. Fig. 10a), output stationary (cf. Fig. 10b), and no local reuse [50].

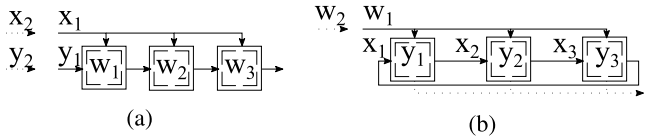


FIGURE 10. Systolic data flow with (a) weight stationary and (b) output stationary adapted from [12], each box represents an individual PE.

Kung *et al.*'s description of data flows focuses on the PE, whereas the perspective can be extended to different levels of the multi memory hierarchy architecture (e. g. [49]). Utilizing different levels of the memory to replicate data and allowing smart data flow and distribution (cf. Fig. 10a) [48], [49], [51] achieves extraordinary low off-chip number of accesses and with this an immense energy reduction. This technique reduces, in combination with batch processing, memory access for CONV layers for AlexNet from 6.7 MB for one image to 15.4 MB / 4 for $B = 4$ [49], [51] to 10.4 MB / 16 for $B = 16$ [48]. The inset in Fig. 9 reveals the benefit of batch processing in combination with an optimized data flow as [49]–[51], represented via the blue dashed line, compared to the minimum off-chip access, represented via the red dotted line. Furthermore, the point 'C [49]a' with off-chip access reaches almost the corresponding red dotted line, yet it stays beyond its theoretical limits, indicated by the blue dashed line, which is exploiting batch processing. This gap can be caused by the overhead in logic for the optimized data flow.

3) COMPRESSION

Off-chip energy is a product of transmitted bits and energy per access. Having optimized both, count and word lengths, as described above, additional savings can be achieved by further reducing the transferred bits through compression. The original classification accuracy is preserved, if lossless compression is applied. Compression exploits the inherent sparsity in activations and weights. On the one hand, sparsity might refer to a low number of non-zero weights and activations. On the other hand, clustered value distributions also lead to a sparse use of the available value range. Thus, it is not surprising that this technique works well with weight sharing [57] to further reduce the total data, which needs to be accessed off-chip and on-chip.

Compression can either be applied to weights [47], [57], feature maps [49]–[51], [53], or both [55], [58]. The most common compression techniques include the compression with sparsity map and non-zero value list [53], the Run-length compression (RLC) [49]–[51], and the compressed sparse column format (CSC) [47], [55], [57], [58]. RLC is a standard compression technique used in communication networks. It captures only the change in symbols in the raw data, i. e. for binary '0' to '1' or vice versa. So, this compression technique is especially useful for long sequences of identical symbols. A good example for the effect of RLC compression combined with data flow optimization is [49]–[51]. It utilizes the fact that the input feature maps consist 57.53 % of zeros. The CSC format was first introduced by [59]. Here, the values are first read column by column out of the target matrix. Then all non-zero values are stored with their value, row, and column index. If the data can be processed in compressed form, further energy E and time-to-solution T reduction is possible, in comparison to processing decompressed data [55].

4) ZERO-SKIPPING

Energy is not only consumed via data movement but also via computation. One technique to reduce the number of computations is zero-skipping, i. e. the MAC is not executed if the input parameter (weight or activation) is equal to zero. Because just non-zero values are saved in compressed matrix it is not surprising that zero-skipping is mainly used in combination with compression. Literature mostly considers zeros in the activations [49]–[51], [53], [57]. Only for [55], [58] both zeros in weights and activations are skipped.

Zero-skipping causes a slight increase in area, due to the additional logic. This increase could be best mitigated by exploiting the statistics of the zeros in weights and with this, reducing the total number of PEs. But the more common technique, utilizing the statistics of the activations, would cause for images with few zeros an increase in time-to-solution. Hence, it would lead to a trade-off between area A and worst case time-to-solution T.

TABLE 2. Benefit of zero-skipping incl. compression in the case of AlexNet.

Reduction of	Total DNN [55], [58]	Only FC layer of DNN [57]
Reduction of MAC to own baseline	6.6 ×	29 ×
Compression factor to own baseline	3.3 ×	10 ×
Reduction of memory req.	3.2 ×	10 ×
Reduction of E wo off-chip access	3.8 ×	29 ×
Reduction of E w off-chip access	3.3 ×	168 ×
Reduction of T	2.7 ×	1 ×
Reduction of accuracy	0.87 %	-

An example for the benefits of zero-skipping and compression is given in Table 2. Both designs process data in compressed form. Energy E and time-to-solution T are more improved for FC layers [57] than for the total DNN [55], [58]. Taking AlexNet as an example, it was shown [55], [58] that for a quantized model the number of total MACs and non-zero parameters can be reduced by more than 12.9× and 100×, respectively. Exploiting such sparsity is non-trivial due to the non-continuous data streams. Ideally, the total computational energy and speed-up scale proportionally to the MAC reduction. The dedicated hardware architecture Eyeriss V2 realizes a speed-up via compression of 2.7× (from 102.1 to 278.7 inferences / s) while energy without off-chip access was reduced by a factor of 3.8×, whereas ideal scaling proportional to the number of zeros would result in a 6.6× reduction. We assume that zero-skipping is also used for the non-compressed AlexNet. It is remarkable that the off-chip access for the compressed AlexNet is reduced by a factor of 3.2× compared to the non-compressed AlexNet, which is almost as good as the expected optimum of a compression factor of 3.3×. The total accuracy reduction of less than 1 % is striking.

As before, energy includes the on-chip energy and the minimum off-chip energy, if the on-chip memory cannot fit the total DNN on-chip. If only the energy without off-chip access is taken into account, the compressed network reduces energy by 29×, cf. Table 2 [57]. But this work is optimized for a tenth of the original memory requirements, thus, the compressed FC layer do not need any additional off-chip access. So, the off-chip access is only added to the non-compressed network, reducing the total energy of the compressed DNN by 168× compared to the non-compressed DNN. Because this design does not skip clock cycles via look ahead function but only gates the unnecessary MACs, the time-to-solution is not reduced. Finally, it is fair to say that the maximum benefit of zero-skipping and compression goes hand in hand with the overall chip design including data flow and on-chip memory size.

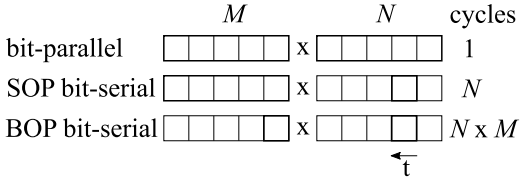


FIGURE 11. Bit-parallel and bit-serial processing of two words with word length, *M* and *N*, respectively.

5) DATA PROCESSING

Data can be processed in either serial or parallel fashion (cf. Fig. 11), with parallel processing being most common in ASIC designs. If the quantization of the input parameter is variable, hardware utilization can be improved by serial processing. Hence, serial processing is mostly used for DNN models with layer-specific quantization of weights or activations. Serial processing either implies both operands (BOP) are used in bit-serial fashion or only a single operand (SOP) is used in bit-serial fashion and the other in parallel fashion.

BOP bit-serial in context of DNNs means that both weights and activations are serially processed as in [34]–[36]. SOP bit-serial processing is applied in [24], [25], [37].

The possibilities for data processing are depicted in the classification tree in Fig. 12, with leaves containing examples for each class. All examples are hardware accelerators for DNN, which support AlexNet.

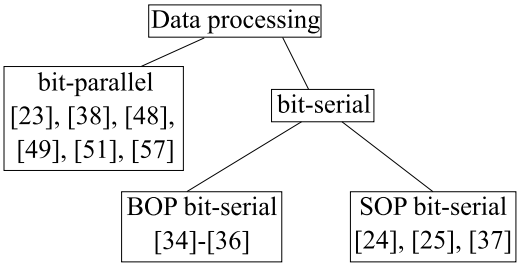


FIGURE 12. Data processing tree with references for AlexNet.

Parallel processing needs more chip area but serial processing needs more processing time, so ideally the AT complexity is constant. Interestingly, bit-serial processing designs are over represented in the Pareto fronts in Fig. 4c despite their, supposedly, less efficient realization, due to requiring additional control and register access. As a side note, the high ATE efficiency of [36], [37] is based on simulated data, only. The most left point in Fig. 3 is [37], whereas in the right dotted circle are the designs from [24], [25], [34], [35]. It is striking that [37] outperforms the other bit-serial designs for the normalized comparison Fig. 3. One reason is its high number of MAC in its DNN, but even the unnormalized AT complexity of [37] is between 16× (for [24], [25]) to 74× (for [34], [35]) lower, cf. Fig. 4c. The area of [37] is 2.6× smaller than for [34], [35]. Furthermore, [37] achieves a 28× lower time-to-solution T compared to [34], [35]. One key reason is the use of a 3D SRAM in [34], [35], whereas [37] uses a classic huge on-chip SRAM. So, [37] needs less area and offers a lower latency. Another major difference between [37] to [24], [25], [34], [35] is its data flow. Unlike [24], [25], [34], [35], [37] does not keep the partial sum in the MAC unit stationary, but uses a fan-in of the partial sums outside of the individual MAC units (this is the inter PE level). So once again, a fan-in based data flow is proven beneficial.

6) CMOS TECHNOLOGY NODE

Digital designs are made to operate in a deterministic fashion independent of manufacturing technology. The choice of technology node and supply voltage plays a key role in setting energy consumption, achievable time-to-solution, and area requirement. Whereby, the voltage scaling potential is also a function of the technology node. The latter drives NRE (non-recurring engineering and tape-out) design costs, i. e. the smaller the node - the higher the price, leading to a rather complex optimization problem.

Comparing design techniques as opposed to the availability of modern process nodes, all digital designs are scaled to a 65 nm node operating at 1.1 V supply voltage. The impact of scaling is visualized for the case of AlexNet in Fig. 13 highlighting the change on the example of energy E.

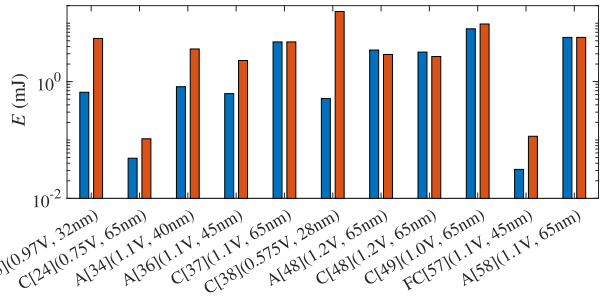


FIGURE 13. Energy E is shown in blue vs. scaled values in red (examples: AlexNet). It is indicated whether the CONV layer (C), the FC layer (FC) or all layers (A) of AlexNet are processed. The original voltage and technology node are given in brackets. All data is scaled to 1.1 V and 65 nm technology node.

7) VOLTAGE UNDERSCALING

Voltage underscaling means the reduction of the supply voltage below its nominal value. It provides a quadratic saving on energy but causes for digital designs frequency reduction (cf. [49] in Fig. 9). Here, the supply voltage and clock frequency scale from 'C [49]a'- 'C [49]c'. The supply voltage decreases only by 1.17× and 1.43× for 'C [49]b' and 'C [49]c' compared to 'C [49]a'. Whereas the clock frequency decreases by 1.30× and 2.65×, respectively.

The data is scaled, cf. Sec. III-A, to a common level, to balance out the energy benefit. But this scaling does not counter balance the frequency benefit. So, even with the total system energy and the related off-chip access, the data point 'C [49]a' is more efficient than its corresponding points 'C [49]b' and 'C [49]c' because the design seems optimized for the supply voltage of the superior point, 1.17 V and with this, its superior clock frequency.

Voltage scaling is not limited to digital designs, but can increase efficiency for analog designs as well. A highly insightful 'ceteris paribus' principle can be found for [28], [29] Fig. 4b, which decreases in 0.2 V steps the supply voltage from 1 V to 0.6 V. This decreases the processing frequency and increases the energy efficiency, so, all three design points are Pareto points, Fig. 4b. The persistently high efficiency comes with an accuracy drop of 0.3 % for 0.6 V.

8) DIGITAL COMPUTING-IN-MEMORY

Classic computer architectures separate computation and storage. The benefits of these designs are their flexibility and split paths for integration and optimization, while having the downsides of inefficiency in area A, time-to-solution T, and energy E caused by a high communication overhead for transfers between cores, memory, and periphery. For the classic von-Neumann architecture adopted in [38] and [47], this explains their high AT complexity.

CIM resolves these drawbacks. In digital designs, deterministic computations are either mapped to an extended bitcell, executed as part of the read-out as in [23] or in a nearby PE tightly coupled to the memory as in [32], [33]. The latter, [32], [33], belongs to the domain of near-memory

TABLE 3. Summary of design techniques with related impact on energy *E*, area *A*, time-to-solution *T*, error *e*, the used dataset for the references which use the technique are MNIST (M), CIFAR 10 (C), and ImageNet (I).

Technique	Dataset	Reference	Impact on			
			E	A	T	e
CMOS Technology Node	M, C, I	[22]–[25], [30], [32]–[38], [42], [46], [47], [53], [55], [57], [58]	↓	↓	↓	→
Off-Chip Memory Standard	C, I	[23], [35]	↓	→	↓	→
Voltage Underscaling	I	[28], [29], [49]–[51]	↓	→	↑	→
Sufficient On-Chip Memory	M	[39], [40], [45], [46]	↓	↑	↓	→
Data Flow Optimization	I	[48]–[51], [55], [58]	↓	↑	↓	→
CIM Analog & MS Design	M, C	[31], [39], [40]	↓	↑	↓	↑
CIM Digital Design	M, I	[23], [32], [33]	→	↑	↓	→
Analog & MS Design Style	M, C	[26]–[29], [31], [39], [40], [45]	→	→	→	↑
Compression	I	[49], [51], [53], [55], [57], [58]	↓	↓	↓	→
Word Length Reduction	M, C, I	all	↓	↓	↓	↑
Small DNN Model	M, C, I	all	↓	↓	↓	↑
High Batch Size	I	[24], [25], [36]–[38], [48], [49], [51], [57]	↓	↑	↓	→
Zero-Skipping	I	[49], [51], [53], [55], [57], [58]	↓	↑	↓	→
Bit-Serial Processing	I	[24], [25], [34], [36], [37]	↓	↓	↑	→
Spiking Architecture	M, C	[30], [45]–[47]	↑	→	→	↑

computing, where as the former, [23], utilizes the same data flow for analog and MS CIM designs described in Fig. 6. The total system energy for CIM in digital IC is diverse from 1.15 pJ / bit as in [32], [33] to 88.5 pJ / bit as in [23].

C. SUMMARY OF TECHNIQUES

The quantitative assessment of design techniques in (Sec. IV-A and Sec. IV-B) is qualitatively summarized in Table 3 via arrows. A down arrow ↓ indicates a decrease of the metric, the up arrow ↑ an increase, and for the right arrow → no impact. The techniques are sorted on the one hand according to their positive and negative impact on cost, and on the other hand according to their focus either on the physical entry level or on the logical and higher levels. Techniques with logical and higher level focus can improve existing general purpose platforms such as, GPU, CPU, and FPGA. We assessed the positive and negative impact on energy E, area A, time-to-solution T, and error e for each technique individually by application of the 'ceteris paribus' principle.

The potential impact is larger for techniques with hardware focus than with algorithmic focus. But the entry barrier for the former is higher than for the latter. For example, the least beneficial technique is processing DNNs on spiking hardware architecture. This fact is especially the case for spiking architectures, which were not designed with the objective to process DNNs. Contrarily, the two most beneficial techniques are the choice of the newest digital IC design technology and off-chip memory technology, with their major drawback being monetary cost.

Also two different versions of the same technique, CIM, can have significantly different impact on efficiency and classification accuracy. Whereas CIM in analog designs has an impact on classification accuracy, CIM in digital designs has none. However, CIM in analog design seems more energy efficient than CIM in digital designs. Finally, both options experience a slight area increase compared to non-CIM designs.

Certain combinations of techniques allow symbiotic effects by either reusing the same preconditions or even amplifying the possible positive impact on the *ATE* efficiency. Thus, these techniques are usually combined. The most common combinations are described in the following.

Typically CIM and analog design style, as well as CIM and MS design style, are commonly implemented together, because CIM naturally reuses preconditions given by the analog and respectively, the MS design style. A sufficient on-chip memory means that all data fit on the on-chip, and the chosen DNN to enable this technique has typically only a few parameters to allow an optimization of area and energy efficiency. Both combinations do not qualify for realistic benchmarks such as ImageNet because they reduce the classification accuracy significantly, thus, they are only used for small benchmarks.

The following three combinations exploit the architecture of bigger DNNs, typically used for more realistic benchmarks. The first one combines data flow optimization and the use of a high batch size to increase the reuse of weights. Hence, off-chip access can be minimized. The second combination also targets the reduction of number of off-chip accesses: the use of low word length combined with compression and zero-skipping. A low word length causes lower precision and more truncation of data, and thus, the data contains more zeros. These can be utilized for compression, inducing fewer off-chip accesses, and for zero-skipping, leading to less MAC. But the word length has a major influence on the classification accuracy. So, further optimization can be reached via a flexible optimized quantization scheme. Therefore, the last combination includes layerwise optimized quantization, which varies from DNN to DNN, and bit-serial processing. The layerwise quantization can be chosen such that the classification accuracy is not negatively impacted. Bit-serial processing allows a full exploitation of the

beneficial impact of layerwise quantization on classification accuracy, while optimizing *ATE* efficiency.

V. CASE STUDY

The variety of design techniques allows designers to operate in a vast design space. However, choosing the right combination of techniques can be rather time consuming. This section explores the design space for a given baseline design, [48], which offers not only high *ATE* efficiency, but also a detailed and transparent documentation that facilitates its implementation. The key feature of this design is the extensive use of optimized data flow comparable to the techniques applied in [49]: data flow optimization and a high batch size of 16 images. However, in [48] not all possible orthogonally applicable techniques are explored. The design space exploration is limited to techniques with no impact on the classification accuracy. The impact of the techniques on *ATE* efficiency is assessed on the basis of the analysis in Sec. IV. To illustrate the complete design space, we also considered the impact on the design if no data reuse techniques are applied.

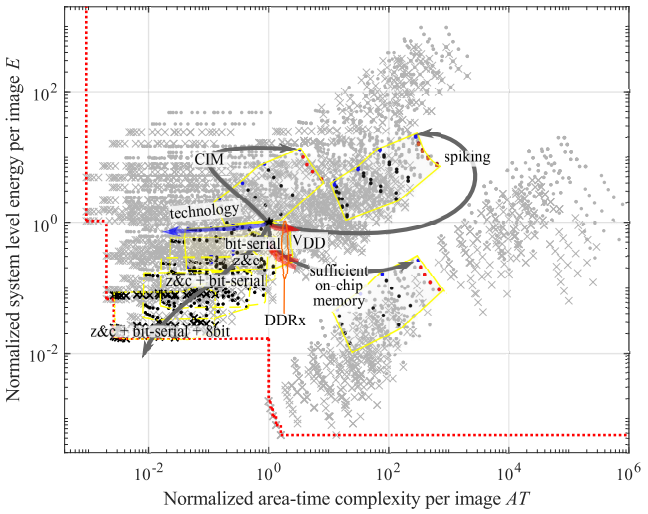


FIGURE 14. Application of optimization techniques without accuracy reduction on [48] as case study, normalized to the original design of [48] processing all layers and including off-chip access.

All design points are normalized according to the scaled baseline design of [48] (65 nm technology and 1.1 V supply voltage), which processes all layers (see Fig. 14). Hence, this data point with normalized energy E and AT complexity equal to 1 is highlighted with a black star. The blue arrow visualizes the impact of the technology node. The blue data points indicate a technology node between 65 nm, 45 nm, 32 nm and 20 nm. The red arrow shows the impact of the supply voltage. For the exploration of each technology node, we use 100 % to 60 % of its nominal supply voltage according to [19]. The red data points demonstrate the different supply voltage steps for the 65 nm technology node. The orange circle around the three different red lines shows the impact of the off-chip memory standard, here abbreviated as DDRx. An alternative to storing data off-chip is the use of a sufficient on-chip memory, which enables storing all data on-chip (grey arrow to the right). We apply the same color code to the mapped design points to facilitate finding the direction of voltage and technology scaling. The applications of CIM (upwards) also assumes a sufficient on-chip memory, so no off-chip memory standard is applied for these design points. The original design uses 16 bit for activations and weights. It is commonly accepted that a word length reduction to 8 bit does not decrease the classification accuracy. The 8 bit data points are denoted by '×' instead of '.' (16 bit data points). The additional orthogonal techniques like bit-serial processing, zero-skipping and compression (z&c) and spiking are highlighted as yellow areas. The red dotted line highlights the resulting Pareto front.

The application of spiking and CIM do not improve the baseline design. Hence, these techniques are not considered in the following. A sufficient on-chip memory reduces energy by 306× compared to the baseline design with DDR3 and increases area by 19×. The trade-off between *AT* complexity and energy E does not encourage a further exploration into the application of a sufficient off-chip memory for this particular design baseline.

TABLE 4. Techniques with no impact on accuracy and their effects on the metrics area *A*, time-to-solution *T* or energy *E*.

Technique	New value	Condition	Metric	Change
Voltage Underscaling	0.66 V (60 % of $V_{DD,nom}$)	65 nm	T	2.25
			E	2.76
CMOS Technology Node	20 nm	$V_{DD,nom}$	A	10.6
			T	2.06
			$E_{on\text{-}chip}$	5.56
Off-Chip Memory Standard	DDR4	65 nm, $V_{DD,nom}$	$E_{off\text{-}chip}$	4.67
			E	2.38
Compression & Zero-Skipping	-	65 nm, $V_{DD,nom}$	T	2.73
			$E_{on\text{-}chip}$	3.80
			E	3.28
Bit-Serial Processing	-	65 nm, $V_{DD,nom}$	T	1.90
			E	1.50
Word Length Reduction	8 bit	65 nm, $V_{DD,nom}$	A	2.47
			T	1.36
			$E_{on\text{-}chip}$	4.48
			E	2.34

A quantitative comparison of the various design techniques, without impact on the classification accuracy, can be seen in Table 4. The *new value* for the used *technique* determines the applied change compared to the baseline. The baseline is as always the CMOS technology node 65 nm, with 1.1 V as 100 % nominal supply voltage $V_{DD,nom}$, 16 bit word length and DDR3 as off-chip memory technology. The *condition* indicates which baseline techniques are not changed, for instance the technology node or the nominal supply voltage for a given technology node. Finally, the last column indicates the increase in efficiency as multiplicative *change* in the corresponding key *metric*.

In total, a Pareto point is created by means of the application of all these orthogonal techniques: technology scaling to 20 nm, voltage scaling to 60% of the supply voltage, off-chip memory technology scaling to DDR4, bit-serial processing,

zero-skipping and compression, and word length reduction. This Pareto point is highlighted in black as the case study in Fig. 3 and Fig. 4c, whereas it is scaled and normalized in the same way as the other data points in the respective figure. In comparison to the baseline point, this scaled Pareto point decreases the total time-to-solution by 7.05×, the total area by 2.47×, and the total energy, including off-chip access, by 11.51×. The application of techniques creates a trade-off between AT complexity and energy E efficiency. Hence, further combinations of techniques create the complete Pareto front in Fig. 14.

VI. CONCLUSION

This article assesses hardware accelerators for DNNs in regard to their specific cost functions. The designs include highly convoluted techniques to achieve the highest efficiency for the lowest accuracy drop. The systematic normalization and scaling of data points of relevant literature enables the dissection of convoluted techniques and their assessment according to their impact on classification accuracy and efficiency in costs as well as in terms of energy per image, area, and time-to-solution. The presented findings significantly improve the design of hardware accelerators and help designers to converge to an optimal Pareto front. Furthermore, this work emphasizes the importance of normalizing and scaling designs to ensure their comparability. Future work could apply similar methods to an analysis of RNN accelerators.

REFERENCES

[1] J. Chen and X. Ran, "Deep learning with edge computing: A review," in *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Jul. 2019.

[2] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[3] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, *arXiv:1605.07678*. [Online]. Available: https://arxiv.org/abs/1605.07678

[4] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural Comput. Appl.*, vol. 32, no. 4, pp. 1109–1139, Feb. 2020.

[5] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNN inference on FPGAs: A survey," 2018, *arXiv:1806.01683*. [Online]. Available: http://arxiv.org/abs/1806.01683

[6] J. Hoffmann, O. Guzman, F. Kästner, B. Janßen, and M. Hübner, "A survey on CNN and RNN implementations," in *Proc. 17th Int. Conf. Perform., Saf. Robustness Complex Syst. Appl. (PESARO)*, Venice, Italy, 2017, pp. 33–39.

[7] B. Li, J. Gu, and W. Jiang, "Artificial intelligence (AI) chip technology review," in *Proc. Int. Conf. Mach. Learn., Big Data Bus. Intell. (MLBDBI)*, Nov. 2019, pp. 114–117.

[8] A. Boutros, S. Yazdanshenas, and V. Betz, "You cannot improve what you do not measure: FPGA vs. ASIC efficiency gaps for convolutional neural network inference," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–23, Dec. 2018.

[9] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides, and M. Shafique, "Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges," in *Proc. ISVLSI*, Jul. 2019, pp. 553–559.

[10] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[11] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

[12] H. T. Kung, "Why systolic architectures?" *Computer*, vol. 15, no. 1, pp. 37–46, Jan. 1982.

[13] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to DNN accelerator evaluation," in *Proc. ISPASS*, Mar. 2019, pp. 304–315.

[14] *The Mnist Database of Handwritten Digits*. Accessed: Jul. 1, 2020. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[15] *The CIFAR-10 Dataset*. Accessed: Jul. 1, 2020. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[16] *ImageNet*. Accessed: Jul. 1, 2020. [Online]. Available: http://www.image-net.org/

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[19] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

[20] T. Stadtmann, C. Latotzke, and T. Gemmeke, "From quantitative analysis to synthesis of efficient binary neural networks," in *Proc. IEEE ICMLA*, 2020.

[21] G. Gielen and W. Dehaene, "Analog and digital circuit design in 65 nm CMOS: End of the road?" in *Proc. Design, Automat. Test Eur.*, vol. 1, Mar. 2005, pp. 37–42.

[22] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1474–1479.

[23] L. Jiang, M. Kim, W. Wen, and D. Wang, "XNOR-POP: A processing-in-memory architecture for binary convolutional neural networks in wide-IO2 DRAMs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.

[24] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[25] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[26] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *Proc. VLSI Circuits*, Jun. 2018, pp. 141–142.

[27] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[28] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 222–224.

[29] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Oct. 2019.

[30] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, Oct. 2016.

[31] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," 2018, *arXiv:1811.04047*. [Online]. Available: http://arxiv.org/abs/1811.04047

[32] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, T. Kuroda, and M. Motomura, "BRein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C24–C25.

[33] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 w," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[34] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 216–218.

[35] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: A 7.49 TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, Jan. 2019.

[36] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 764–775.

[37] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–12.

[38] G. Desoli, N. Chawla, T. Boesch, S.-P. Singh, E. Guidetti, F. De Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh, and N. Aggarwal, "A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 238–239.

[39] Z. Jiang, S. Yin, M. Seok, and J.-S. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 173–174.

[40] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–490.

[41] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Jun. 2015, pp. 92–104.

[42] C.-H. Tsai, W.-J. Yu, W. H. Wong, and C.-Y. Lee, "A 41.3/26.7 pJ per neuron weight RBM processor supporting on-chip learning/inference for IoT applications," *IEEE J. Solid-State Circuits*, vol. 52, no. 10, pp. 2601–2612, Oct. 2017.

[43] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.

[44] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, no. 7, pp. 1601–1638, 1998.

[45] F. N. Buhler, P. Brown, J. Li, T. Chen, Z. Zhang, and M. P. Flynn, "A 3.43TOPS/W 48.9pJ/pixel 50.1nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm CMOS," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C30–C31.

[46] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640M pixel/s 3.65 mW sparse event-driven neuromorphic object recognition processor with on-chip learning," in *Proc. Symp. VLSI Circuits (VLSI Circuits)*, Jun. 2015, pp. C50–C51.

[47] C. Liu, G. Bellec, B. Vogginger, D. Kappel, J. Partzsch, F. Neumärker, S. Höppner, W. Maass, S. B. Furber, R. Legenstein, and C. G. Mayr, "Memory-efficient deep learning on a SpiNNaker 2 prototype," *Frontiers Neurosci.*, vol. 12, p. 840, Nov. 2018.

[48] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2220–2233, Aug. 2017.

[49] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 262–263.

[50] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 367–379.

[51] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Nov. 2017.

[52] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2012, pp. 37–48.

[53] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, "NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 644–656, Mar. 2019.

[54] M. Schaffner, F. K. Gürkaynak, A. Smolic, and L. Benini, "DRAM or no-DRAM? Exploring linear solver architectures for image domain warping in 28 nm CMOS," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, May 2015, pp. 707–712.

[55] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[56] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmaeilzadeh, "SnaPEA: Predictive early activation for reducing computation in deep convolutional neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 662–673.

[57] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254.

[58] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Understanding the limitations of existing energy-efficient design approaches for deep neural networks," in *Proc. SysML*, 2018, pp. 292–308.

[59] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE*, vol. 55, no. 11, p. 1801—1809, Nov. 1967.

CECILIA LATOTZKE received the B.S. degree in electrical engineering from Friedrich-Alexander University Erlangen-Nuernberg, Erlangen, Germany, in 2014, and the M.S. degree in electrical engineering from RWTH Aachen University, Aachen, Germany, in 2017, where she is currently pursuing the Ph.D. degree.

Her current research interest includes efficient deep neural network acceleration on FPGA as well as on ASICs, with regards to influence on classification accuracy.

TOBIAS GEMMEKE (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from RWTH Aachen University, Germany, in 1998 and 2006, respectively.

In 2004, he transitioned to IBM's Research and Development Organization, Böblingen, Germany, targeting high-performance processors. In 2007, he joined former start-up Aquantia Corporation (now Marvell) conceiving energy efficient PHY solutions for the 10GBase-T over copper standard. In 2011, he became the Technical Lead of the Digital Design Team, Holst Centre/IMEC, Eindhoven, The Netherlands, focusing on ultra-low-power design of wearable sensor nodes. Since 2017, he has been a Full Professor with the Chair of Integrated Digital System and Circuit Design (IDS), RWTH Aachen University. His research interest includes the design of digital systems considering all entry levels from algorithmic optimization down to the physically oriented design.